a2 addition, it is shown being fed by the pricing solution when it may only receive flight information from the scheduler process 16 depending on the airline.

The paragraph beginning on page 7, line 7, is amended as follows:

a3 The client system 30c receives the results from the server process 15. These results are the set of pricing solutions 38 and/or pricing solutions based upon availability. The client process 36 executed in the client 30c uses this information or a subset of it to access a booking system 62 to provide a booking and reservation for a user selected, enumerated pricing solution, as will be described below.

The paragraph beginning on page 8, line 5, is amended as follows:

a4 The client process 36 receives the flight information from scheduler process 16 and the pricing solution from the faring process 18 or the availability system 58 and enumerates pricing solutions from the directed acyclic graph (DAG) representation. The enumerated set of pricing solutions is rendered in a graphical user interface 41 on the client monitor 40 (FIG. 1) in a manner as will be described below.

The paragraph beginning on page 8, line 12, is amended as follows:

a5 In response to user input 76, the client 30c can manipulate travel options and can query the local copy of the DAG to produce and display a subset of pricing solutions enumerated from the DAG that satisfy the query 76. The manipulation process used to control the display and change the travel options will be described below.

The paragraph beginning on page 8, line 18, is amended as follows:

a6 A directed acyclic graph (DAG) is used to represent the compact set of pricing solutions 38' since, in general, the number of nodes needed to represent a typical pricing solution will be substantially less than the actual number of pricing solutions represented by the DAG. This significantly increases the efficiency of transfer of a set of pricing solutions 38 from the server process 15 to the client process 36. The DAG representation also minimizes the storage requirements for the set of pricing solutions 38. The DAG representation permits the use of

a6 cont'd

powerful search, sorting and manipulation processes to produce various subsets of set of pricing solutions in an efficient manner. As used herein, a directed acyclic graph (DAG) is a set of *nodes* connected by directed arcs, that have no loops of arcs in the same direction. If a node A is connected to a node B via an arc A6B, then A is called a parent of B, and B is called a child of A. Each node may have zero, one or many parents and zero, one or many children. As used herein, a pricing solution that is represented by a graph will be referred to as a pricing graph.

The paragraph beginning on page 10, line 29, is amended as follows:

a7

The set of pricing-solutions represented in the pricing-graph is presented in TABLE 2 below.

The paragraph beginning on page 15, line 14, is amended as follows:

a8

After the faring process 18 decomposes the itineraries into faring atoms, the faring process 18 retrieves fares 84 and rules 86 for each faring atom by accessing the fares/rules database 20a mentioned above. At this point a fare's routing is retrieved from a routing database and applied to a faring atom. If the routing test fails, the fare cannot be applied to the faring atom and a fare component is not built.

The paragraph beginning on page 15, line 21, is amended as follows:

a9

The faring process 18 applies the rules 88 to the faring atoms to produce fare components. Fare-components are combinations of faring-atoms and fares. Fare-components (TABLE 6) are produced if a fare's rules pass a preliminary check on a faring-atom. They are used to store deferred rules (e.g., deferred record-2s and combinability record-2s) that are applied at a later stage of processing. Fare components also store extra information produced during the rule-checking process, such as information about surcharges and penalties and discounts that are applied to the base fare price.

NE.

In TABLE 6 on page 16, please substitute " ' " for all occurrences of "=".

The paragraph beginning on page 16, line 10, is amended as follows:

From the fare components the faring process 18 constructs 90 priceable units. For certain types of rules such as those which require access to fares and/or flights from outside of the fare component, those rules are stored 88a in the fare component for later or deferred evaluation. The priceable unit process 90, takes valid fare components and constructs priceable units from the fare components. This process 90 involves grouping fare components from different slices and checking fare component combination restrictions. At this stage of processing, the rules deferred in steps 88 and 88a are reapplied.

a10

The paragraph beginning on page 17, line 14, is amended as follows:

After evaluation of the deferred record-2s at the priceable unit stage, the itineraries and priceable units are grouped together into a complete set of pricing solutions. This occurs by a link process 94 that links itineraries to corresponding pricing units from different slices to provide the pricing solution. At this juncture, any remaining cross-priceable unit fare combinability checks are performed to eliminate invalid combinations.

a11

The paragraph beginning on page 18, line 26, is amended as follows:

The pricing solution resulting from the linking process 94 is used to construct 96 a pricing graph from the various data structures built during the preceding processes. This pricing graph is transmitted to the client process or can be stored for later use or transmission. A pseudocode representation of the high level processing logic involved in the above search procedure is set out below in TABLE 11.

a12

The paragraph beginning at page 19, iine 5 has been amended into two paragraphs as follows:

Referring now to FIG. 5, the process 82 to decompose an itinerary into faring atoms includes a retrieval process 100 that retrieves all itineraries for all slices in a journey. For each itinerary in each slice, the process 82 groups fairing atoms by faring markets at 104 and partitions itineraries into the divisions of faring atoms at 106.

a13

A pricing graph 38' (FIG. 3) is produced containing logically combinable nodes that can be used to efficiently and compactly represent a set of pricing solutions 38 (FIG. 1). The pricing

a13 Cont'd graph 38' as used herein is a so-called directed acyclic graph although other types of representations could be used. For example, a grammar could be used.

The paragraph beginning on page 20, line 5, is amended as follows:

The pricing graph 38' is constructed from data structures (summarized below in TABLE 12 and mentioned in conjunction with FIGS. 4A-4B) provided during the preceding processes. The data structures convert to one or more nodes in the pricing graph. The open-label-set data structure becomes an OR node and its children are the converted versions of its backward links. Each backward link in the open label set is converted to an AND node. If a a14 pricing object is shared, for example, a priceable unit core is shared between several priceable unit labels, then its counterpart nodes in the pricing graph will be shared so that the size of the pricing graph is not unnecessarily enlarged. The converted nodes are placed in the pricing graph nodes. Terminal objects such as fares and itineraries undergo essentially no conversion. They are placed into special terminal nodes with no children and are obtained from the various data structures that have the pricing objects.

The paragraph beginning on page 23, line 3, is amended as follows:

The pricing graph 38' resulting from the search procedure provides a compact way for representing a very large number of set of pricing solutions. By the above process, it is often possible to obtain a very large number of pricing solution components. Although the number of pricing solutions can be returned in-the form of a simple list, this is not desirable, as a very large a15 number of pricing solutions can be difficult to manipulate, enumerate and interrogate and to transfer/transmit across a network since the amount of data involved is very large. The pricing graph 38'provides a more compact way of representing these pricing solutions. The compact representation of the range of set of pricing solutions is generated where choices are represented explicitly and redundancy is removed wherever possible.

The paragraph beginning on page 24, line 4, is amended as follows:

As mentioned above, the pricing graph 38' produced by the search procedure includes a16 three types of nodes. The first type of node is a node that represents choices called "LOGICAL

a16 Cont'd OR" nodes. The second type of node is a node that represents collections referred to as "LOGICAL AND" nodes. A third type of node represented in the pricing graph is a terminal node that represents pricing objects.

In TABLE 14, on page 24, please substitute " ' " for all occurrences of "=".

The paragraph beginning on page 25, line 9, is amended as follows:

a17 Referring now to FIG. 6, a high level illustration of a process 300 that operates on the pricing graph 38'typically as a client process 36 on the client computer system is shown. The process 300 includes a user query 302 that passes parameters into a process 306 and a value function 304 to extract from the pricing graph 38' certain pricing solutions that satisfy parameters specified by the user query 302. The extracted pricing solutions are returned as a list or other representation. Generally the pricing solutions are displayed on the display 40. Display software handles production of GUI's 41 to present the information.

The paragraph beginning on page 25, line 20, is amended as follows:

a18 The pricing solution list will contain pricing solutions extracted from the pricing graph 38'in accordance with user specified parameters from the user query 302 using one of the processes 304 of FIG. 7, where some exemplary illustrative processes are shown. In particular, in response to the user query 302, one of the processes is executed. The processes 304 can comprise a find best "value" and pricing solutions associated with the value (e. g., price) process 304a; find best "value" and pricing solutions associated with the value for "node" (e. g., find best price for a particular itinerary) process 304b; an enumeration for "N" pricing solutions 304c; or an enumeration process that lists the pricing solutions according to some "value." Additional enumeration processes can be provided to produce query results corresponding to different ways of looking at pricing solutions extracted from the pricing graph 38'. A node invalidating process (not shown) that invalidates selected nodes from contributing to a pricing solution is also included.

The paragraph beginning on page 26, line 7, is amended as follows:

Efficient algorithms 304 are used for manipulating this representation to extract information of interest and to enumerate set of pricing solutions from the structure. For example, it is possible to quickly extract the cheapest solution; to find the cheapest solution involving selected fares and itineraries; to verify whether any pricing solution remains if specific fares or itineraries are excluded; to enumerate solutions under various orderings and so forth. Furthermore, the representation is compact enough so that it can be efficiently stored and transmitted such as from the server to the client. One benefit, therefore, is that after a single fare search in the server process 15, the server process 15 transfers the pricing graph 38 to the client process 36. The client process 36 can examine and manipulate the large space of pricing solutions represented in the pricing graph 38' without further interaction with the server process 18.

Q19

The paragraph beginning on page 26, line 23, is amended as follows:

For the set of pricing solutions represented by the pricing graph 38'to be useful, processes are provided to extract pricing solutions from the graph and manipulate the set of pricing solutions, as depicted in FIG. 7. In general, each of the enumeration processes to be described operate on the pricing graph to extract pricing solutions from the pricing graph according to particular criteria that can be set, for example, by a client system 30c (FIG. 2) in response to a user query 48 (FIG. 2). Examples of user queries as well as a display representation for information extracted from the pricing graph will be described below.

Q20

The paragraph beginning on page 27, line 21, is amended as follows:

There are many processes or operations on the pricing graph 38' that use a *value-function,* a function that operates on the terminal nodes of the pricing graph 38' and returns a numerical value that can be used to rank pricing-solutions. Examples of 25 value-functions include *price* computed by summing the prices of fares (and penalties and surcharges) in a pricing-solution, duration, or convenience (that might be a mix of total travel-time with penalties for stops and airline-changes, for example), or mixes of each.

Q21

The paragraph beginning on page 32, line 2, is amended as follows:

Another procedure 304b finds, for each node, the best (i.e., minimum) value of some value-function over all the set of pricing solutions involving that node. Price function 304b finds for each itinerary, the cheapest price of any pricing solution that contains that itinerary. These values can be computed efficiently, if the value-function can be decomposed into a node-value-function.

The paragraph beginning on page 32, line 9, is amended as follows:

The best price value function 304b computes inner-values, as above, and computes for every node, an outer-value, equal to the minimum value contributed by all parts of the graph except that represented by the node. For each node, the minimum value of the value-function over all solutions that involve the node, (i.e., the total-value) is computed as the sum of that node's inner-value and outer-value.

The paragraph beginning on page 33, line 3, is amended as follows:

It may be desirable to "invalidate" certain nodes from the pricing-graph 38'. For example, itineraries containing or not containing specified airlines could be marked as not participating in the above algorithms, enabling the algorithms to find the best solutions involving or not involving these itineraries. The above algorithms can be easily adapted to accommodate checking whether the node is valid. In particular, the computation of inner-values, the first step in all the above algorithms, is modified to mark for every node whether the node represents any valid partial pricing-solutions given a specific query parameter. This information can be used in the rest of the algorithms. Every terminal node contains a field "valid?" that is either true or false. The compute-inner-values procedure uses these values to set the "valid?" field for non-terminals. See TABLE 17 below:

The paragraph beginning on page 41, line 7, is amended as follows:

The diversity process 360 thus includes a procedure for generating a diverse list of (N) travel options (Rts) from a larger list of travel options (Ts), that are the best travel options for a set of travel requirements (R), as defined by an ordering function F. The diversity process 360 generates 362 an prioritized (ordered) list of requirements Rs, and sorts 364 the list of travel

options (Ts) by function (F) to produce a best-first ordered list (Ts2). The diversity process 360, initializes the list of result travel options (RTs) to be empty. If the remaining list of requirements (Rs) is empty, the process 360 returns an ordered list of diverse travel options (Rts). Otherwise, the diversity process selects 366 the first travel requirement (R) from the ordered list of requirements (Rs) and removes 368 a requirement (R) from the requirement list (Rs). The diversity process 360 finds 370 a first (e.g., best) option T in the best-first ordered list (Ts2) that satisfies travel requirement (R).

a25 Cont'd

The paragraph beginning on page 41, line 24, is amended as follows:

In some embodiments the set of travel options is represented by a data structure that stores a large set of travel options by representing permitted combinations of smaller travel option components such as airline flights and fares. In such cases the travel option selection process above is implemented using a more complicated operation than searching through an ordered list. With the pricing-graph the process for finding 370 the best travel option that satisfies a travel requirement is implemented for a representation that expresses travel options in terms of permitted combinations of smaller travel option components by disabling option components inconsistent with the requirement.

a26

The paragraph beginning on page 42, line 3, is amended as follows:

As shown in FIG. 9A, the process 360 can use the node invalidating functions to invalidate nodes 422 in the pricing graph 38 that are inconsistent with the requirements. The process 360 applies 424 an enumeration algorithm that extracts the best solution from the pricing graph representation. The diversity process 360 calls an enumeration function, as described above, to enumerate all of the valid pricing solutions from the pricing graph that are remaining after the diversity process selectively invalidated nodes in the graph inconsistent with the travel requirements.

a27

The paragraph beginning on page 42, line 13, is amended as follows:

If no option in the best-first ordered list (Ts2) satisfies 372 the requirement (R), the process 360 goes to check 374 if the remaining list of requirements (Rs) is empty. Otherwise, the diversity process determines 376 if a travel option T is not already in result travel options list

a28

A28 Cont'd

(RTs). If the option T is not in the list (RTs), the diversity process adds 378 the travel option T to end of the result travel option list (RTs). If the size of the travel option list (RTs) is equal to or greater than N 380 the process returns the ordered list of diverse travel options.

The paragraph beginning on page 42, line 23, is amended as follows:

A29

Referring to FIG. 10, the diversity process 360 could optionally determine 382 for a travel requirement (R2)in the set of travel requirements (Rs), whether the requirement (R2)is included in a prior requirement (R), and whether the travel option T also satisfies 384 the requirement (R2). If the travel option T satisfies the requirement (R2), the process 360 can remove 386 the requirement R2 from the requirement list (Rs) and return to determine 374 (FIG. 9) if the remaining list of requirements is empty.

The paragraph beginning on page 43, line 19, is amended as follows:

A30

The large candidate set of travel options may be analyzed 394 to find all parameters e. g., airlines found in any travel option, all departure dates for outbound and return, and all departure parts-of-day (morning, afternoon, evening) for outbound and return. The ordered list of requirements is generated by filling 396 in for each template all airlines, dates and parts-of-day present in the options.

The paragraph beginning on page 44, line 1, is amended as follows:

A31

Referring to FIG. 12, an alternative diversity process 400 to generate a diverse set of travel options from a larger set of travel options is shown. The alternative diversity process 400 generates the best one or more travel options as defined by each of a set of different travel preference functions. The alternative diversity process 400 defines a set of travel preference functions with each function capable to order travel options. In one example, the set might include "cheapest", "quickest", and "most-convenient" where each is a function that assigns a numerical score to a complete travel option (such as price, total-trip-time, and total trip-time with penalties for stops). Functions that assign numerical values based on combinations of cost and convenience are possible, such as functions that weigh both price and time.

The paragraph beginning on page 44, line 15, is amended as follows:

Given set of travel options Ts 401, a set of preference functions Fs, and a desired number of answers for each preference function Ns, the alternative diversity process 400 returns a reduced set of diverse travel options Rts. The alternative diversity process initializes 402 a list of result travel options RTs to be empty and for each preference function F in the set of preference functions Fs and number of travel options (N) in the set of desired number of answers in each preference function (Ns), the alternative diversity process 400 computes 404 the N best travel options in Ts as defined by F. For each travel option T, unless the travel option T is in the set of diverse travel options Rts 406, the alternative diversity process 400 adds 408 the travel option T to the set of diverse travel options Rts and checks 410 the number of options. The alternative diversity Process 400 outputs 412 the diverse set of travel options (RTs).

A32

The paragraph beginning on page 44, line 30 is amended as follows:

The diversity process can be run more than once with different travel option preference functions (a set of F's). For example, a travel planning system may wish to output a diversity of travel options that include diverse options that are cheap and diverse options that are convenient, reflecting uncertainty in whether a traveler is cost-sensitive or convenience-sensitive.

A33

In the claims:

Please amend claim 1 as follows:

SUB B1

-- 1. A travel planning system comprising:

a requirements generator module configured to generate a plurality of travel requirements;

a selection module configured to output a set of diverse travel options smaller than a candidate set of travel options by selecting from the candidate set of options, for each travel requirement in the plurality, one or more travel options that satisfy that travel requirement, wherein the candidate set is represented by a compact representation.--

A34

Please add claims 3-26.